An Integrated Approach to Process Planning and Scheduling Using Genetic Algorithms

Philip Husbands

PhD.

University of Edinburgh

1993

Contents

List of Figures

List of Tables

Abstract

This thesis presents a new integrated approach to process planning and job-shop scheduling. The relationship between planning and scheduling is reassessed and the line between the two tasks is made significantly more blurred than in the usual treatment. Scheduling is traditionally seen as the task of finding an optimal way of interleaving a number of fixed plans which are to be executed concurrently and which must share resources. The implicit assumption is that once planning has finished scheduling takes over. In fact there are often many possible choices for the sub-operations in the plans. Very often the real optimisation problem is to simultaneously optimise all the individual plans *and* the overall schedule. This thesis describes how manufacturing planning has been recast to allow solutions to the simultaneous plan and schedule optimisation problem, a problem traditionally considered too hard to tackle at all. A model based on simulated coevolution is developed and it is shown how complex interactions are handled in an emergent way. Results from various implementations are reported.

Underlying this new approach is a feature based process planning system that is used to generate the space of all possible legal process plans for a given component. This space is then searched, in parallel with spaces for all other components, using an advanced form of genetic algorithm. The thesis describes the development of the ideas behind this technique and presents in detail the constituent parts of the whole system.

Declaration

I hereby declare that this thesis has not been submitted, either in the same or different form, to this or any other University for a degree. It was composed by me and describes my own work.

Acknowledgements

Many thanks to Frank Mill, the project supervisor, for giving me the opportunity to work on these problems. Frank made a number of very important suggestions throughout the course of this work. The general idea. of separating the plan reasoning and plan optimisation stages was Frank's. He also suggested investigating the use of genetic algorithms. He provided support and encouragement on many different, levels and I count myself lucky to have worked with him.

Thanks are also due to Stephen Warrington who was my co-worker on the SERC project of which this research formed part. Stevie taught me a lot about the practicalities of manufacturing and developed all the knowledge and data bases used in this work. If Stevie hadn't been so easy to get on with I'm sure the work would have gone much less smoothly.

I acknowledge useful conversations with many other colleagues at home and abroad. They are too many to mention, but in particular I'd like to thank Stephen F. Smith of CMU Robotics Institute for some very helpful comments on an earlier paper to come out of this research.

Last but not least, many thanks to Alison for putting up with the disruptions to family life while I put this together.

Chapter 1

Introduction

1.1 Introduction

Modern manufacturing is a complex business involving many disparate functions. Any large manufacturing company will have whole divisions dedicated to, for instance, marketing, accounting and high-level strategic planning. However, it might be claimed that at the heart of many manufacturing enterprises is the three stage process of designing the products; planning how to manufacture the products; and deciding how to concurrently manufacture several different products so as to make best use of the production facility. The first of these activities, unsurprisingly, is referred to as design engineering, the second as process planning and the third as manufacturing scheduling. Over the past two decades, or so, more and more advanced computer based methods have been developed and introduced into both design and manufacturing. In the field of computer aided design (CAD) we have systems, to mention only a few, capable of drafting designs, modelling 3D components, or calculating stress concentrations over a given model. In computer aided manufacturing (CAM), as well as computer controlled manufacturing machines, there are computer based tools for aiding in, or sometimes automating, process planning, scheduling and other areas of manufacturing planning such as material handling (delivering materials to the right place at the right time). Present CAD/CAM systems

have yet to realise their full potential for industrial users. This is largely because of the lack of integration of the various packages used in most installations. Recently there has been much support for the argument, that advanced computer aided process planning (CAPP) systems should play an important role in bridging the gap between CAD and CAM [19, 108, 55].

In most manufacturing environments. for a generative process planning system[1] to fulfil1 its potential, a number of key issues must be tackled. Two of these issues have been concentrated on here. Firstly. the degree to which the planning system is integrated into the overall production system. Secondly, the planner should not just produce feasible plans but optimal or near optimal plans according to some global costing criterion.

The research described in this thesis centres around the development of a prototype generative process planning system which was designed with these two issues firmly in mind. The system is for use in a. multi-machine environment and deals with conventional metal removal processes on prismatic components. More specifically, the integration issue was tackled by allowing the planning system to be very closely linked to a design system, and by developing planning techniques which allowed a more flexible approach to scheduling. Close regard to the problems of communication with a computer based design system led to the adoption of feature based methods in the planning. That is, every component is thought of as being composed of a number of manufacturing features (holes, slots, pockets,..) to be cut from the blank. These features are related by a network of geometric and technological constraints.

The optimisation issue was tackled by taking a radical new approach involving the use of stochastic search over a space of all possible process plans. This approach was extended to tighten the integration between planning and scheduling by severely blurring the traditionally rigid line between the two.

[1] 'A generative process planning system produces plans from scratch given a description of the part to be manufactured and knowledge of the manufacturing facility. This will be discussed in more detail in the next chapter.

The vast majority of the research described in this thesis was carried out in the' Department of Mechanical Engineering, University of Edinburgh during the period 1986-1989 as part of a SERC(ACME) project entitled Representation, *Reasoning and Decision Making in Process Planning with Complex Components*. Only those parts of the research carried out by the author are described in detail; those parts which were done collaboratively (some of the work described in Chapter 3) are clearly indicated. Much of the material reported here has been published elsewhere, in slightly different form, as papers in various journals and international conference proceedings.

1.2 Traditional Approaches to CAPP

Because of the complexity of the task, generative CAPP was an obvious area for the application of AI. Most of the recent work on this topic has involved the use of AI techniques [26, 79, 83, 59, 111]. As already mentioned, any practical planner should consider the most efficient way to manufacture the part. The criteria to be used will vary according to the nature of the manufacturing facility but are likely to involve interaction with other areas of the overall manufacturing planning task, such as scheduling. This aspect of the problem is important because there are often vast numbers of alternative ways to manufacture a component. These will vary in the order and number of operations, the choice of machines and tools and workpiece orientation. Very often these alternative plans will have widely differing costs. The orderings and operation choices will usually be subject to various constraints, many of which are generated as part of the planning process. All this adds up to a very formidable problem.

Much early work ignored this dimension of the problem, but those researchers concerned with generating efficient (low cost) plans have tended to follow traditional AI or OR approaches [26, 111, 81]. These involve using heuristic search to greatly reduce the size of the problem space considered. The heuristics are embedded within the planning algorithm and a single plan is gradually built up as a solution to the

problem. Finding sufficiently powerful heuristics is often extremely difficult. Because of the complexity of the problem, these sorts of approaches are very unlikely to find anything like a global optimum in anything but the simplest of cases [105,13]. However, they avoid generating very inefficient plans.

It is very difficult, to find a general way to take into account interactions with other component's process plans with this sort, of approach. Hence the planning and scheduling functions are usually treated as separate. The scheduler's job is to interleave, as efficiently as possible, the separately generated process plans for some number of components to be manufactured concurrently. Some recent approaches have generated some alternatives in the plans to give the scheduler more flexibility [106, 102].

The method described in this thesis, which amounts to a new planning paradigm, relies on the implicit generation of a huge number of possible plans. This space of plans is then searched for the optimal solution.

1.3 A New Approach to CAPP

This thesis concentrates on two core aspects of a complete framework for dealing with a certain class of design and manufacturing problems. The overall approach is now briefly presented. This is captured, at a very high level, in Figure 1.1. A design system, whose description is outside the scope of this thesis[2] produces manufacturing feature based component and blank representations. These representations are compared in order to find out which component features are to be machined and which, if any, already exist in the blank. The complete space of plans for each component is implicitly generated, giving all the ordering and operation parameter

[2]When this work was started the design system did not exist, but its interface to the plan generation system was defined, so we could act as if it did. This system has now been developed and is linked to the modules described here.

alternatives.[3] These spaces are searched in parallel, taking into account interactions between and within plans, using an ecosystem model, based on an advanced genetic algorithm. From this emerges a. solution to the simultaneously optimal plans and schedule problem. That is, the plans for each component to be manufactured are individually optimised according to such criteria as machining costs and setup costs; but the optimisations are done in parallel with interactions taken into account. Hence the plans are individually optimised as much as possible while causing as few interactions (e.g. bottle necks in overall schedule) as possible. There is no explicit scheduling stage, but, a. schedule emerges, for a schedule is just a description of the parallel operation of a number of plans. This new technique replaces the traditional two stage planning-then-scheduling approach and effectively re-evaluates the relationship between planning and scheduling: in this new view they are inextricably part of the same problem. The earlier, knowledge based, parts of the system (plan space generator) determine the boundaries and structure of the search space that the emergent optimisation techniques work in (parallel genetic algorithms). This approach makes very heavy use of genetic algorithms, a. powerful search technique very loosely based on natural evolution, a topic which will be dealt with in detail in a later chapter.

The last two modules of this system are dealt with in this thesis, for further details of other aspects of the system see [57, 30].

1.4 Thesis Contributions

The research methodology used was to develop prototype systems aimed at proof-of-concept demonstrations of general techniques. As such, the main contributions of this thesis are as follows:

[3] This refers to the fact that all the data needed to explicitly construct the search space point by point is made available. This amount of data is of course quite manageable, whereas the explicitly generated search space would certainly not be. Enumerative search on this kind of problem is quite out of the question.

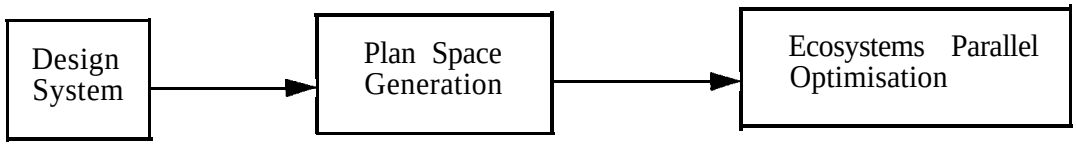


Figure 1.1: Overall approach.

- The development of general algorithms and representation techniques for generating spaces of all possible process plans for a class of prismatic parts;
- The analysis of the resultant optimisation problem of searching this space for a single near-optimal plan;
- The development of a useful cost function to allow the application of optimisation techniques to this problem, the function has a number of subtleties which are conducive to more efficient search;
- An investigation into the application of traditional search techniques to this problem;
- The development of a genetic algorithm to search this space in a far more satisfactory way than the previous techniques;
- An experimental investigation into the use of this technique on the process plan optimisation problem;
- The development of a sophisticated extension of this technique to handle the parallel optimisation of many process plans for different components, thereby tightly integrating planning and scheduling and effectively re-evaluating the

job-shop scheduling problem by showing how the standard definition is far more restrictive than necessary;

- Several of the points above also resulted in general contributions to the field of genetic algorithms:
  - An early use of complex integer strings;
  - An early use of heuristics with the genetic search;
  - The first ever use of a 'multi-species' parallel distributed genetic algorithm.

1.5 Thesis Outline

The thesis is structured as follows.

Chapter 1. Introduction

Chapter 1 is this introductory chapter.

Chapter 2. Background

This chapter provides some technical background to material covered later in the thesis. Process planning is discussed and various approaches to computer aided process planning are outlined. There is a short section giving an overview of optimisation and search since these topics loom large in this work. Other research in CAPP is reviewed, relating it to the approach developed in this thesis. To allow a better flow, other technical background material, such as that on genetic algorithms, is introduced later in the appropriate parts of the thesis.

Chapter 3. The Plan Space Generator

This chapter describes the plan space generator: that part of the system used to find the space of all possible process plans for a given component. The methods for representing information about the component and blank, the machine shop, and the manufacturing practices to be followed, are described. The algorithms for using this

information to generate the plan space are detailed. Two different implementations of the system are discussed. A number of research issues and limitations are brought, out.

Chapter 4. The Process Plan Optimisation Problem

This chapter analyses some general properties of the search spaces generated by the plan space generator. Methods are developed for estimating the size of a particular instance of the problem. A simple cost function suitable for use with most combinatorial optimisation techniques is introduced. A far more subtle version is developed! this second function is shown to be conducive to more efficient search.

Chapter 5. The Application of A* and Branch and Bound

The earliest attempts at discovering a solution to the process plan optimisation problem were based on using general heuristic search. Because of the complexity of the problem, it was decided that this was the most promising area of 'conventional' optimisation techniques. The techniques used were adaptations of the *A** algorithm and the branch and bound method. This chapter describes how these techniques were applied and presents the results obtained.

Chapter 6. Application of Genetic Algorithms to the Process Plan Optimisation Problem

The experiments with conventional search techniques described in chapter 5 were not very successful. This led to an attempt to develop a technique that worked with complete candidate plans. Genetic algorithms were considered suitable and were successfully applied in a robust and general way. At the time this research was done they were a very obscure technique and had never been applied to a problem of this scale. This part of the research, and that of the next chapter, make a number of contributions to the field of genetic algorithms as well as to CAPP and manufacturing scheduling. Since the genetic algorithm parts of the research described in this thesis are probably the most original, and provide its major contributions, in order that the reader may follow the technical material, this chapter provides an introduction to genetic algorithms. The application of the technique to process plan optimisation is then described. Results from a number of experiments are presented

and a. series of specially developed heuristic search methods for seeding the initial population are described.

Chapter 7. An Ecosystems Model for Integrating Planning and Scheduling

Being able to find individually near optimal process plans may be of very little value if there is no communication between the planning systems and the machine shop scheduler. A number of individually optimal plans may interact to cause serious bottle-necks in the schedule. This chapter develops a way of facilitating the necessary integration of planning and scheduling to avoid this problem. The technique, based on multi-population parallel distributed GAs, is capable of simultaneously optimising the process plans of a number of components while taking into account interactions between them. At the same time a near-optimal schedule emerges. This radical new approach to integrating process planning and scheduling amounts to a. re-evaluation of the job-shop scheduling problem, by pointing to a. more general and fundamental characterisation than the one normally used. Two different implementations of the basic idea are compared. Results from a number of experiments are presented. The limitations of these preliminary experiments are discussed and future research directions are highlighted.

Chapter 8. Conclusions

This chapter sums up the contributions of the thesis and rounds up with some general conclusions.

Chapter 2

Background

2.1 Introduction

This chapter provides some technical background to material covered later in the thesis. Process planning is discussed and various approaches to computer aided process planning are outlined. An overview of optimisation and search is given since these topics loom large in this work. Other research in CAPP is reviewed, relating it to the approach developed in this thesis. To allow a better flow, other technical background material, particularly that on genetic algorithms, is introduced later in the appropriate chapters of the thesis.

2.2 Process Planning

A process plan is a detailed set of instructions on how to manufacture a given part. Typically it lists the order in which operations must be performed, the appropriate machine tool to use for each operation and the required machining parameters. A more general definition is given by Chang and Wysk [12]:

> Process planning is that function within a manufacturing facility that establishes which machining processes and parameters are to be used (as well as those machines capable of performing these processes) to convert (machine) a piece part from its initial form to a final form predetermined (usually by a design engineer) from an engineering drawing.

A process planner must take into account constraints dictated by both the part geometry and required tolerances when choosing appropriate machines, processes, tools and setups (part orientation on the machine bed), and imposing an order on the operations.

In addition a process planner must select jigs and fixtures. These are devices for guiding a tool or holding a workpiece in a position most suitable for machining. Very often these devices are non-standard and their proper use is a skilled business. In general a process planner will have to perform the following tasks:

- Work piece selection
- Manufacturing Process selection
- Process equipment selection
- Cutting tool selection
- Setup selection
- Fixture and fitting selection
- Operation sequencing
- Process parameter selection
- NC instruction generation

The last of these refers to generating the code to run computer controlled cutting machines.

2.2.1 Computer Aided Process Planning

Manual process planning requires a great, deal of time, knowledge and experience. Skilled planners take many years to build up the required experience. Automated process planning has obvious attractions.

Early computer based process planning systems used editors and other tools to help in report and plan sheet generation, and in the storing and retrieval of plans [107]. Such systems can greatly improve the efficiency of a planner and many are still in use.

More elaborate CAPP systems can be divided into two categories, variant and generative. Variant planning relies on data, retrieval procedures to find standard plans for similar components. In order to do this, group technology codes are used to describe designs in terms of geometric and manufacturing features. However, there are many drawbacks involved in the use of such codes: tedious data. entry, ambiguity when the code is too short, over specialization when the code is too long. Of course the method is of no use when the component to be planned does not fit into any of the standard part families. Generative planning attempts to overcome these problems by building a plan from first principles for each part. Such a system requires a detailed description of the part and of the manufacturing facility.

The research described in this thesis falls under the heading of generative process planning, although it will be seen that the approach taken is highly non-standard.

The form in which the component is represented as the input to a CAPP system can be a crucial issue, heavily effecting the potential of the system. Generative systems normally use some sort of special description language which might be based on a CAD format or a solids modelling format, or as in this work, a specially designed symbolic language which describes a part as a collection of features (holes, slots, pockets etc.) and their relationships. This idea seems to date back to Descotte and Latombe [26].

Specific approaches to generative CAPP will be discussed later in this chapter. But, since some of those involve optimisation, and optimisation is absolutely central to this thesis, first there is a short discussion of optimisation and search.

2.3 Optimisation and Search

An optimisation problem involves minimising (or maximising) some cost function. For functions of continuous variables various calculus based methods are often used [5]. One technique is to solve the set of equations resulting from setting the gradient of the objective function to zero (partial differentiation wrt to each variable yields zero). More common are the ideas of hill climbing, for maximisation, or gradient descent, for minimisation. In order to perform gradient descent (hill climbing is directly analogous), choose some starting point, and move downwards in the direction with steepest slope until a minima is reached. Both of these methods are local in scope; they require confinement to some restricted neighbourhood of the point currently under test. Because of this they are prone to missing global minima by becoming stuck in local minima. Figure 2.1 illustrates this point; if a gradient descent had started at points A or B it would have found local minima only, it would have had to have started at, for instance, C to find the global minima. These techniques also require well-behaved functions: the slope must be everywhere defined. Many practical optimisation problems involve noisy discontinuous cost functions. For these problems calculus based methods are rarely robust – it may be possible to tune them to perform extremely well in restricted circumstances, but they usually do not perform adequately over a wide range of problem instances whose exact properties can not be predicted.

Most combinatorial optimisation problems[14, 109] (finding an optimal combination from a set of resources) cannot be cast in the terms required for gradient descent methods. Typically these problems involve discrete units, such as the machines, tools, and setups of the process plan optimisation problem, or the operations of the manufacturing scheduling problem, both central to this thesis, or the cities of the

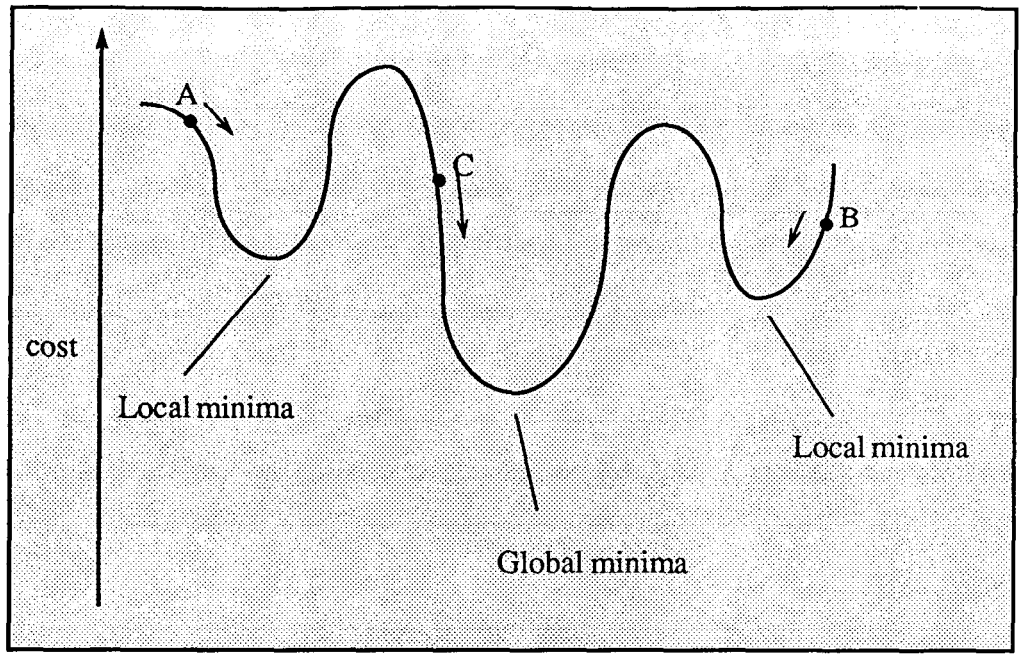


Figure 2.1: Local and global optima.

well known travelling salesman problem [14]. For these sorts of problems various search techniques may be applicable. The notion of a search space is a very powerful one in characterising how these methods work. These are abstract spaces in which each point represents a possible complete or partial solution to the problem. When each point represents a complete solution the space is of a fixed dimensionality and the optimisation proceeds by searching through this space. moving from point to point looking for the lowest cost solution. Where each point represents a partial solution, the concept of dimensionality does not apply and the search proceeds by moving from partial solution to partial solution trying to build up the least cost complete solution. It can also be useful to think in terms of a space of nodes – each node is a step in the solution and the aim is to find a low cost path from some start node to a goal node, passing through a number of intermediate nodes.

Various search techniques exist for tackling these sorts of problems. A large number of enumerative methods have been developed by workers in both Operations Research (OR) and Artificial Intelligence (AI). In their simplest form these methods look at the cost of every point in the space or every potential path to a goal node. In that form they are only of any use on small problems as the search time becomes infeasibly large for bigger problems. For the N city travelling salesman problem there are N! points in the space of complete solutions (permutations of the cities).

This is 125 for 5 cities, 40,320 for 8 cities and 2.7×10^{32} for 30 cities, here we have an example of *combinatorial explosion.* One way of overcoming this difficulty is to use heuristic search. This involves augmenting some enumerative search technique with a heuristic (rule of thumb) that, is designed to guide it to a. good solution while only considering a very small proportion of the search space. If such heuristics can be found the search is then computationally feasible. Finding good heuristics is usually very difficult, especially if some robustness is required. Generally heuristics might be made to work well in one set of circumstances but fall over as soon as they are presented with a. new set. These methods can very often be fooled by local optima.

A third class of techniques are search algorithms which involve the use of randomness. Simple random search (picking points from the search space at random and saving the best) is usually no better than exhaustive search (complete enumeration). However, techniques that involve some stochastic elements. such as simulated annealing [66] and genetic algorithms [49, 37], can avoid the use of heuristics and overcome the curse of local optima. The random elements allow these methods to sample points over a. large volume of the search space and accept poor solutions which may lead to an optimal one. Techniques from the second and third classes will be described in more detail later when their application to the process plan optimisation problem, and a generalised version of the job-shop scheduling problem, is explained.

It should be noted that the search for global optima in large complex spaces is fraught with a major difficulty – very often it is difficult or impossible to judge if we have found a global minima. So usually we are interested in near-optimal solutions, that is solutions which appear to at least be good local minima. This often means finding a technique that consistently performs better on the particular problem than other techniques tried.

2.4 Various approaches to generative CAPP

Most generative CAPP systems perform backwards planning. Assuming that we have a finished component, the goal is to 'fill' it until it matches the starting blank. A drilling process 'fills' a hole, a slot milling process 'fills' a through slot, and so on. Backwards planning starts with the finished form, whose properties are known, and searches backwards for processes to fulfil1 (at least) the worst preconditions that the previous process can start from (e.g. find a roughing process that leaves an intermediate state suitable to start the finishing process chosen for fulfilling the final component requirements). Planning back to less accurate initial conditions generally involves far less search than attempting to plan forwards, making sure all preconditions are accurately satisfied. Backwards planning is the approach taken in this thesis.

Early generative CAPP systems tended to use decision trees to encode the manufacturing knowledge and planning strategies [116]. These systems were very inflexible and difficult to maintain. The restrictions of such approaches led to investigations into the use of AI techniques in CAPP.

Descotte and Latombe's GARI planner is one of the best known early AI based generative CAPP systems [26]. It employed a production rule knowledge base to store process capabilities. The right-hand side of each rule encoded a manufacturing action (or choice) and these were weighted according to how closely they should be followed. Parts and machines were described using special purpose symbolic structures, the part descriptions were feature-based. The core of the planning algorithm used constraint propagation; at each cycle an action was taken to further constrain the solution until a plan had been built. Feature interactions were not taken into account except for very simple cases. However, this was a demonstration of a powerful approach which has been very influential.

Inui et al. [59] were among the first to attempt to integrate CAPP more closely with design. Their expert system based CAPP system used a dynamic feature-

based product model. They used a comparison process and constraint checking procedures not unlike those introduced in the next chapter.

Hayes et al. [46] used a fairly sophisticated expert system approach for planning for a single machining centre. They only dealt with simple components but they did make an attempt to tackle the cost and operation sequencing problems in more detail than the projects mentioned above, all of which ignored the plan cost as a consideration. Hayes et al. embedded a minimise-setup heuristic in their planning algorithm which met with some success.

van't Erve [111] developed a prototype expert system for machine selection and cutting tool characteristics for prismatic parts. Again, cost was taken into account in making choices and sequencing operations. A local best first search was embedded into parts of the decision making machinery.

Gindy [81] describes a hierarchical structure for feature definition and an information structure for developing process plans for prismatic parts. Again some local search has been incorporated into the planning logic to improve plan costs by minimising setups and the like.

Although these later approaches do take into account plan costs to some extent, the motivations for investigating the approach described in this thesis were the difficulties encountered with the more traditional approach of embedding heuristic search into the core planning algorithms. If we characterise the process planning task as, essentially, to find an optimal distribution of [machine/process/tool/setup] combinations over a set of features, there is the added difficulty of taking into account the interactions between the features which impose constraints on the plan. These constraints appear as partial orderings between operations and can often only be discovered during the planning process. If optimal plans are to be found using heuristics embedded in the planning algorithm, these heuristics must be strong enough to handle the combinatorial explosions which are inevitable in this kind of problem, as well as being able to cope with the fact that constraints are being discovered all the way through the planning procedure. That kind of approach did not look at

all promising for the general multi machine environments we wished to model. The FORBIN project at Yale was attempting to solve a similar problem with a traditional approach, a report on their experiences [105] detailed overwhelming problems with the combinatorics of the search.

Hence it was decided to attempt to implicitly generate the entire space of plans, including all constraints. It was felt it would be easier to deal with a. completely defined search space as opposed to a partially defined one.

Palmer and Hall [88] later took a similar approach, using branch and bound search on a space of possible plans. They were not able to produce very good results, having very similar experiences to those described in Chapter 5, where the application of branch and bound search is dealt with.

Vancza and Marcus [110], quite independently, also decided to investigate optimisation within a completely defined space of plans. They also decided to use genetic algorithms. Their approach was quite different to that described later in Chapter 6, using a very different representation, and different genetic operators, and acting on more constrained problems than those investigated here. Nevertheless. they too demonstrated that the use of genetic algorithms in process plan optimisation can be a very powerful approach.

2.5 Scheduling

Scheduling is the task of sharing out the factory resource> (machines, labour, tools etc.) given a set of process plans describing how to manufacture a number of different components. Essentially it consists of interleaving the plans so as to make best use of the resources, assuming that there are potential clashes where the same resource might be needed in the manufacturing of more than one component. Very often the objective is to minimise makespan, that is to minimise the period spent manufacturing the components. The input to scheduling is very often a set of fixed plans, although, as we shall see in the next section, there may be some alternatives

in the plans. Scheduling will be discussed in a more formal way in Chapter 7 which is concerned with a new approach to the problem.

2.6 Integrating Process Planning and Scheduling

As mentioned in the last chapter, a major aim of this thesis is to introduce a technique for integrating process planning and scheduling. There is a huge body of work on schedule optimisation, which is understandable since it is the schedule, the overall picture of how the manufacturing of all components is coordinated, that sets limits on the overall performance of the manufacturing facility. However, there are often many alternative ways of building a process plan. Hence. there is a deep sense in which planning and scheduling can be integrated: they can be regarded as a *single* optimisation problem. If we can find the optimal way of manufacturing each component (i.e. find cheapest process plan) *while* at *the same time* minimising the interactions between the plans (usually the province of scheduling) we will have effectively solved this deeper problem and facilitated the integration. It is this sense of integration that we shall concentrate on in this thesis. However, there are a number of papers which discuss process plan and schedule integration at a much higher, managerial, level, e.g. [68].

There has been very little work on this deeper sense of integration, but that that has been done is reviewed here.

Chryssolouris et al. [15] regard the allocation of factory resources as a common element suitable for the integration of process planning and scheduling. However, they state that process planning and scheduling use quite separate criteria (technological constraints and timing considerations respectively) and hence integration is a matter of viewing the overall problem as one of multi-criteria decision making. The integration they feel is possible is not as profound as the one sought here.

Most other researchers active in this field take a similar view to that held by the author. Namely, certain aspect of the planning are separate from scheduling, and do

use quite different criteria, but the common ground is the objective of minimising manufacturing costs.

The most common approach to some form of integration at the cost level is to generate flexible plans with some decisions left to the scheduling stage; that is, providing some alternatives for the schedule to work with, rather than fixed plans. The usefulness of this sort of approach was recognised as early as 1980 by Halevi [44]. Sundaram and Fu [101] later provided a simple technique in which a small amount of flexibility was introduced into a set of plans to be scheduled. They showed how a more efficient schedule could then be produced.

Krause and Altmann [67], among others, represent alternative in plans by using tree structures. Alternatives are represented on separate branches of the tree. Tonshoff et al. [106] use more efficient graph structures instead; after branching has occurred divergent paths can meet up again at a later common node. They refer to the resulting structures as non-linear process plans. They allow more flexibility by having both OR splits (one of the alternative paths is followed) and AND splits (both of the paths are followed but the ordering is not specified). This provides greater room for manoeuvre at the scheduling stage and results in better schedules. While the motivations were similar, these researchers have only looked at a very small part of the integrated problem presented in Chapter 7; they only have a small amount of flexibility in their plans compared with the approach developed in this thesis. Another major difference is that the technique presented later effectively does away with a separate scheduling stage.

Khoshnevis and Chen [65] carried out research into the development of a good set of priority or dispatching rules for use with flexible process plans, as a way of further facilitating integration.

Liang and Dutta [70] have also pointed out the need for a deep integration between the planning and scheduling problems, but their proposed solution was demonstrated on a very small simplified problem. It is not clear from their descriptions how well their method would scale up to the sorts of problems tackled later in this thesis.

Some years ago Iwata et al. [60] used branch and bound search (see Chapter 5) to tackle scheduling with alternative machines (i.e. some flexibility in the plans) but with no alteration in operation orderings. They found good solutions but only for very small problems. It is highly unlikely that their method can be scaled. Certainly the experiences of using branch and bound for a complex single plan optimisation task, described later in Chapter 5, bear this out. Others have had similarly negative experiences with branch and bound [88].

The only other piece of work I am aware of that attempts to handle problems approaching the complexity of those tackled in this thesis, is work by Palmer [87], inspired by earlier versions of the work presented in Chapter 7, and making use of another stochastic optimisation technique, simulated annealing. Although his techniques have not been applied to such large problems as dealt with later, they do appear promising.

2.7 Summary

This chapter has introduced a number of key topics which have very strong bearing on the subject matter of this thesis. Process planning was described, as were approaches to computer aided process planning. Optimisation and search were briefly outlined. The need for taking the cost of a process plan into account while it is being generated was explained. Recent approaches to generative CAPP which attempt to do just this were discussed. The role of scheduling was sketched out. The importance of integrating process planning and scheduling was brought out and then a review of approaches to this problem was given.

Further related work will be introduced and discussed at appropriate points throughout the thesis.

These first two chapters have now set the scene for the remainder of this thesis which describes in detail the technical research which is its contribution.

Chapter 3

The Plan Space Generator

3.1 Introduction

The new approach to process planning and scheduling developed in this thesis was outlined in Chapter 1. This chapter describes the plan space generator: that part of the system used 1 o find the space of all possible process plans for a given component,

Any generative process planning system must have the capability to reason in detail about the parts to be manufactured. Clearly this will involve having access to a representation of the properties of and interrelationships between the manufacturing features (holes, slots, pockets . ..) of any given component. The planner would also need a global view of the work piece as it is transformed from blank to finished component. Similarly there must be some method available for modelling the machine shop. The author developed, in conjunction with Frank Mill and Stephen Warrington [56, 57, 55], a particular feature based part description language, strongly based on earlier work by Mill [75]. The same method has been extended to model the machine shop, it is described in Section 3.3. The system described in the following sections was developed some years ago, it has recently been replaced by a more efficient, but largely functionally equivalent, object oriented system that will be briefly described at the end of this chapter. The earlier work was done jointly with Frank

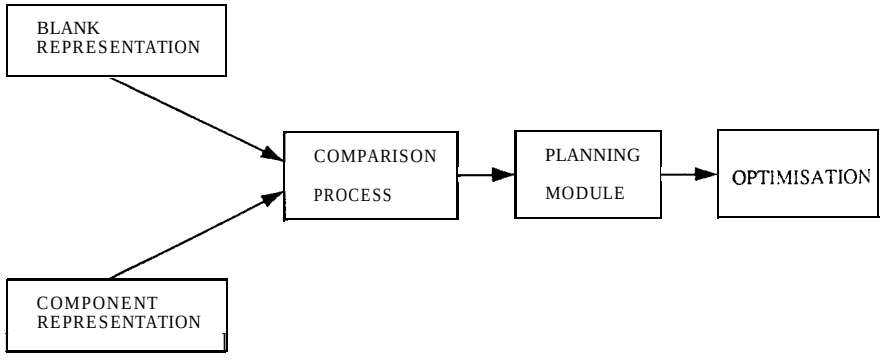


Figure 3.1: Overall approach

Mill and Stephen Warrington. Those aspects which are the author's contributions are focused on here. Frank Mill's work on part representation schemes has already been mentioned, Stephen Warrington worked on the manufacturing knowledge bases and machine shop data bases. The author's contribution was largely in developing the algorithms to manipulate this information, and in devising the representation schemes for much of the knowledge (e.g. the rule base syntax).

The overall approach used is captured, at a very high level, in Figure 3.1.

Representations of the blank and of the component are compared in order to find out which component features are to be machined and which, if any, already exist in the blank. The plan space module generates the complete space of plans which is searched for the best solution at the optimisation stage.

3.2 Overview

The plan space generation algorithm attempts to break the manufacture of a. component down into a number of nearly independent steps. The entire space of possible plans can then be generated by finding all the possible operations to carry out each

step along with the ordering constraints which must exist between the steps. Essentially a. step refers to a finishing operation on a. single feature or super-feature (a group of features treated as one due to some network of constraints binding them together) or a roughing operation on an intermediate feature (defined later). So each step of the plan has a unique feature, super-feature or intermediate feature associated with it. The operations found to manufacture these are described in terms of [machine/process/tool/setup/cost] combinations. The setup refers to the orientation of the workpiece and the cost refers to the machining cost associated with that operation. Along with this information the planner generates a separate network representing the partial orderings it has deduced hold between the stages of the plan.

The simplest way to describe the algorithm in more detail is to start with the highest level structures it builds and manipulates. These aae planning networks like the one shown in Figure 3.2. In common with most generative process planners! the manufacturing processes are treated as material addition operations, whereas of course they actually involve material removal. The overall strategy is to start with those features deepest in the component and work out towards the surface. This process is guided by a set of 'critics' constantly on the look out for possible feature interactions, which may result in deferring work on part of the component [57] , and by high level considerations regarding datums and such like. Once a feature has been chosen, a finishing process to achieve its desired final state is inferred. The details of this are discussed later. This finishing process leaves an intermediate feature with various inexact properties, such as a range of possible surface finishes. This models the fact that most finishing processes can only sensibly be started from a state with a given range of properties. For instance, it is highly undesirable to end up grinding down a very rough uneven surface. A roughing process is then chosen to manufacture the intermediate feature. Remembering that an exhaustive set of possible manufacturing routes is required, for any given finishing process, any number of compatible roughing processes may exist. Thus a network like the one shown in Figure 3.2 is built up, keeping track of the interactions between finishing and roughing processes for each feature. These networks can be readily extended to

allow an arbitrary number of sub-finishing and sub-roughing processes, and hence intermediate features, to be handled. Each route on the network, from starting conditions to final feature, has its own subsidiary information attached, such as machining parameters and cost. In complex cases it may be desirable to weight, the different routes or to remove certain nodes or connections. It is often found, when building up the network, that a possible roughing operation is exactly the same as the finishing operation it is connected to via an intermediate feature. In this case the roughing node and its connections are removed and a connection made directly from the starting conditions to the finishing node. This tells us that it is feasible to machine out the feature using the single process. Various kinds of links between the sub-networks of different features are built up by the planner.

The output from this process is a large number of interconnected networks like the one shown in Figure 3.2. A manufacturing plan for the sub-goal described by the fragment of network shown is a route from the starting conditions node to the goal conditions node. Implicit in the representation are functional dependencies between sub-operations. The algorithm also discovers ordering constraints in the processing of the various features, intermediate features and super-features. This results in the output of a partial ordering graph like the one shown in Figure 3.3. Each of the symbols refers to a particular feature, intermediate feature or super-feature.

The system will now be described in detail.

3.3 Feature-based representation

Much of the reasoning needed in process planning is at the symbolic levei and involves the manufacturing features of the component. Because of this, the most appropriate part description languages for use with CAPP systems are largely symbolic feature based ones. Such a representation should describe the necessary geometric, topological and manufacturing properties of the component. Manufacturing properties mainly refers to the various required tolerances of the part.

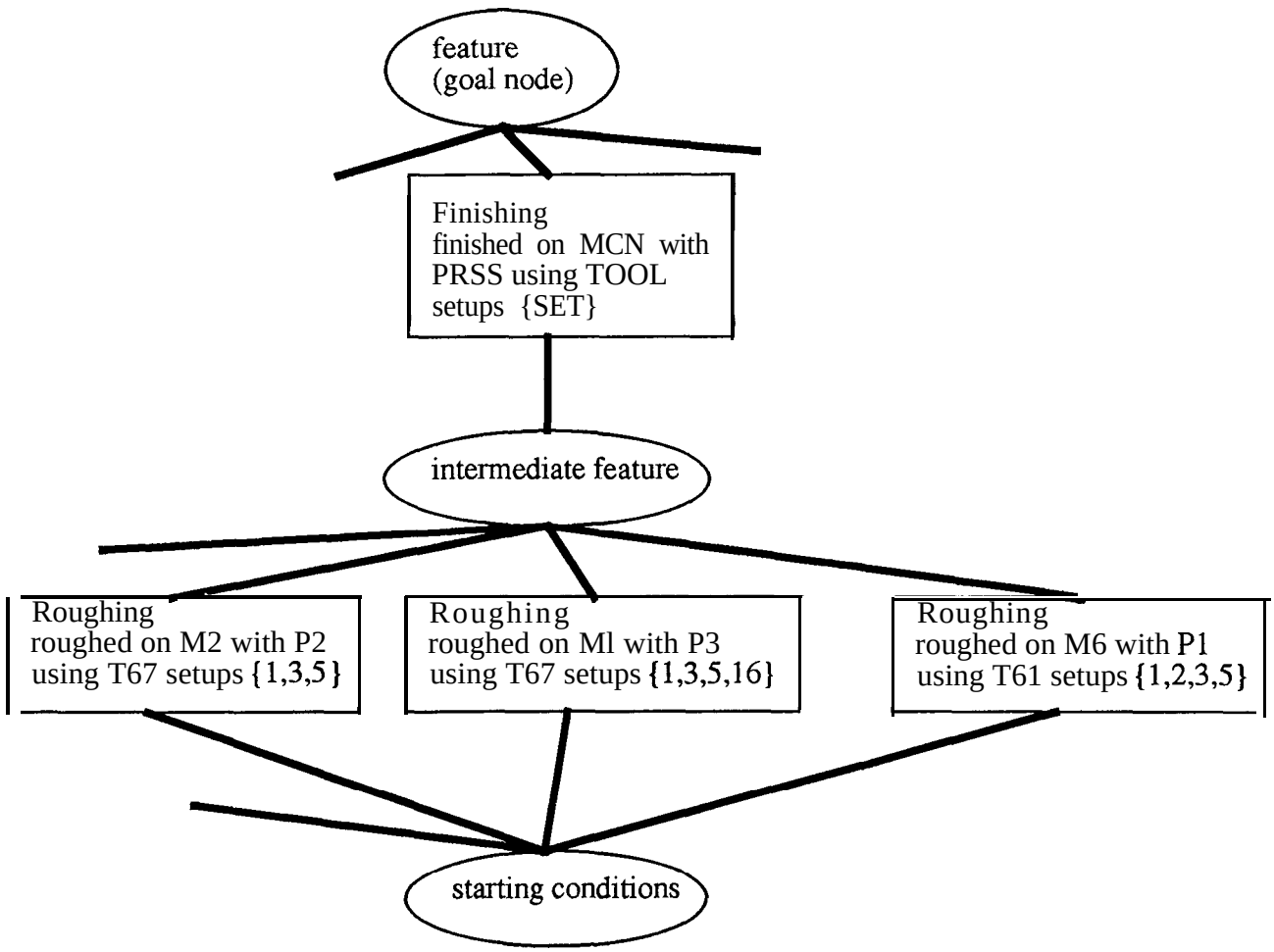


Figure 3.2: Fragment of planning network.

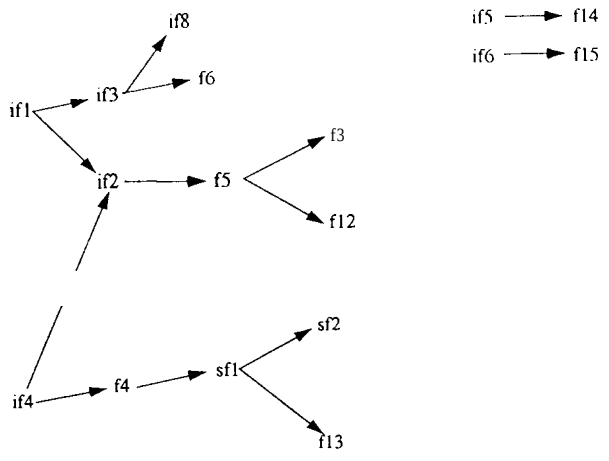


Figure 3.3: Anteriority constraints.

The representation should be complete in process planning terms. For instance, it may not be necessary to describe the whole geometry and topology of a component in order to decide how to manufacture it. A complex surface might be represented in terms of a few parameters, such as maximum radius of curvature, needed to reason about it. On the other hand, it is almost certain that the interrelationships which exist between the features should be described in full. A component should not be regarded as a. collection of independent features - that would imply that a process plan is generated by choosing an operation for the manufacture of each feature in isolation and then compiling these into a list. In fact, most feature interactions impose constraints on the shape of the plan.

The representation should be flexible. It should be a simple matter for the user to add new features or feature properties to the system.

A smooth interface should exist between the manufacturing knowledge base and the part representation data base, avoiding repetition of information.

The same points apply equally well to the machine shop representation.

A network approach was adopted, as it was thought capable of fulfilling the requirements discussed above. In this scheme a component or blank is described by a

binary relational database where each entry consists of a. triple of the form:

<entity> <relation> <entity>

An entity will typically be a feature name, a number: an atom or an uninstantiated variable. The technique allows the use of some higher level relations which treat a triple as an entity itself and hence allow the possibility of chained relations. This is usually employed when an initial relationship needs to be qualified. The method provides a simple way of representing interactions between features. For example, if two faces p1 and p2 are related by a parallel tolerance, this can be written as:

```
p1 para p2 withtol 0.03.
```

The following is a. small part of the representation of the simple test. component shown in Figure 3.4.

```
h2 isa blind-hole.
p5 vexedges p6.
p7 hasfeat pkl .
e17 edges p6.
s1 comprises p10.
p5 para p8 withtol 0.005.
```

The feature types u ed in this work were as follows:

- plane (flat surface)
- through hole
- blind hole
- through slot

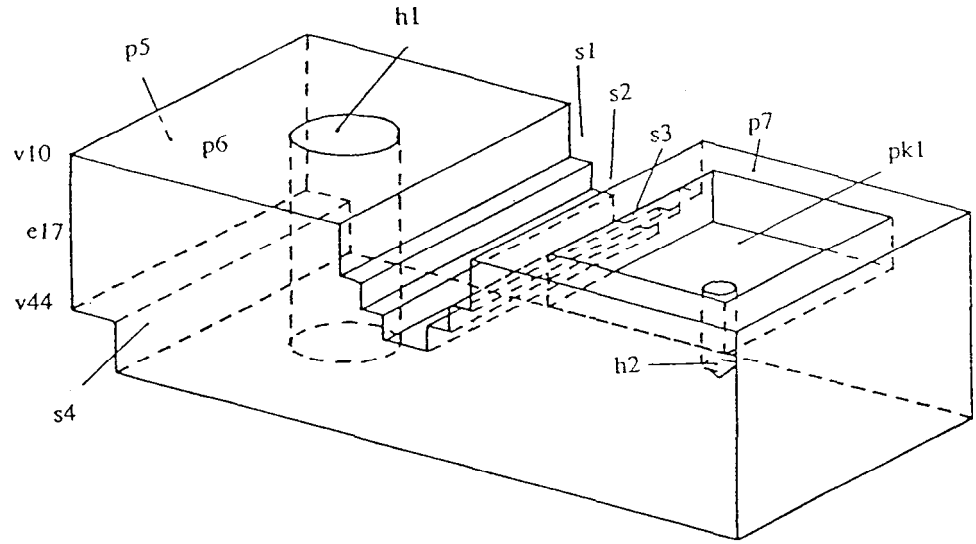


Figure 3.4: Example component

- end slot (step)
- pocket

Components in which the features have simple orthogonal geometric relationships to each other were used. However, complex blanks, such as castings, out of which orthoganally related features needed machining, can be handled. Examples of such components will be shown later.

The whole system was implemented in Edinburgh PROLOG [2] and the relations are defined as infix operators of the appropriate precedence. The whole network is thus represented as a PROLOG file listing the individual relations.

The meaning of the individual relationships shown above should be fairly self-explanatory. Very important relations are 'isa', allowing inheritance within the networks: and 'hasfeat', defining where one feature is a sub-feature of another. The representation is very similar to the semantic networks knowledge representation technique used widely in AI [9, 111]. The same method was also used to model the

machine shop in terms of machine and tool capabilities. Thus parts of machine and tool descriptions could be as follows:

```
drill1 isa drilling-machine.
tool1 isa twist-drill diam 3.
drill1 angutol 1.0 using drilling.
```

Figure 3.5 shows how, by using this method, the part representation and the machine shop model integrate in a very simple and natural way. The use of such a homogeneous description of the basic manufacturing data. simplified the task of building a manufacturing methods knowledge base which straddles the two areas. Indeed it meant, a very flexible and simple interface between the knowledge bases and data bases could be defined. This is discussed in more detail later in the chapter. The machine shop model used in most of the work reported here consisted of 12 machines: 2 drilling machines; 2 slab milling machines; 7 vertical milling machines; and 1 grinding machine. Each of the machines was modelled on a real machine using data from manufacturers' catalogues.

3.4 Comparison of blank and component

The representations of the blank and the part must be compared to ascertain what is to be machined. This is the stage at which the goals of the planner are derived. The method employed is highly feature oriented and is based on a comparison from each of six orthogonal directions. Each feature, both from the component and the blank, is involved in some binary relation giving its relative orientation. For instance, 'pl dvect posx' refers to the fact that a vector perpendicular to the face pl and pointing out of it runs in the positive x direction. After unifying the coordinate systems used to describe the geometry of the blank and the component, this directional information is used to build up a set of directional surfaces (Dsurfs). For example, the posy Dsurf is the set of surface features seen by observing the object along

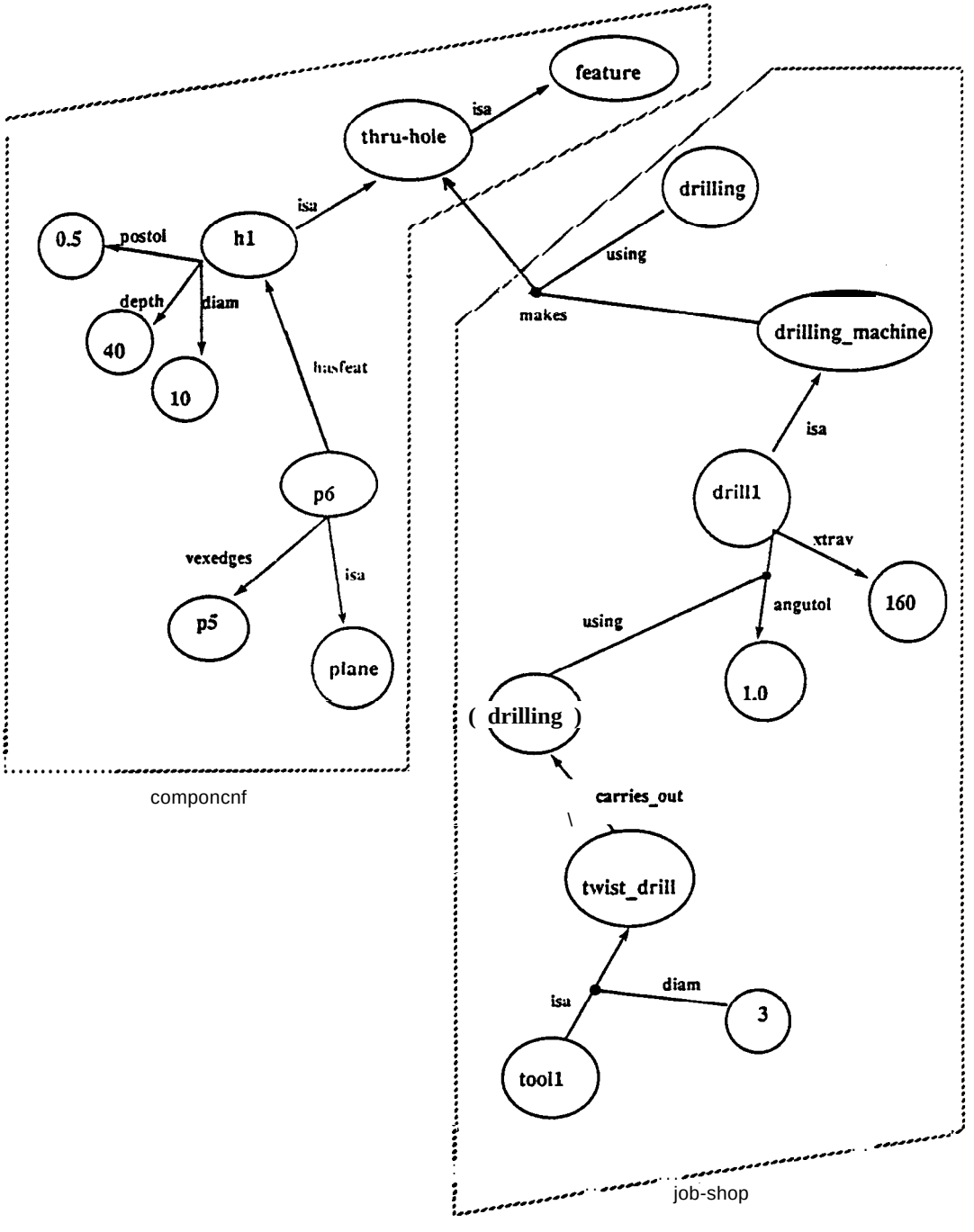


Figure 3.5: Sample of representation network

the negative y direction. Corresponding Dsurfs from blank and component are compared. The reason that this is done, and that Dsurfs are considered at all, is that the part representation method dictates that all other features are sub-features, or sub-sub-features etc., of some surface feature. If any of the surface features needs machining, that is there is no equivalent feature on the blank, then it is assumed that all of its chain of sub-features will also need machining. If this were not the case the features would have to lie within the blank and not be traceable up a 'hasfeat' chain to a surface feature - geometrically impossible given the representation scheme. If a feature is a. sub-feature of one that already exists in the blank, a mapping between the sub-feature and a corresponding blank sub-feature is looked for. The mapping process uses tolerance information, such as surface finishes, as well as geometric information. This is because the feature may exist in the roughed state on a. blank but needs to be finished to produce the required component feature. This may well be the case where the blank is a casting. This comparison method has been found to be far more efficient than an earlier 'brute force' method which exhaustively compared a.11 generic feature types in the component and blank.

The simplest type of blank possible would be like the one shown in Figure 3.6. The posy Dsurf, that is the set of surface features seen by observing from outside the object and along the negative y direction, for this simple object consists of the plane p502. However, for the component shown in Figure 3.4 the corresponding Dsurf consists of the planes p6 and p7, the through-slot sl, the pocket pki and the through-hole hl.

The comparison algorithm is outlined in Figure 3.7. The outcome of this process is a list of features labelled as 'mc-features', that is features in the component that need machining. The comparison process is not necessary in the later C++ implementation of the system as this is linked directly to a design system where the designer builds the component starting from a model of the blank. Hence differences from the blank, i.e. features needed machining, can be automatically recorded during the design process. The later implementation is described at the end of this chapter.

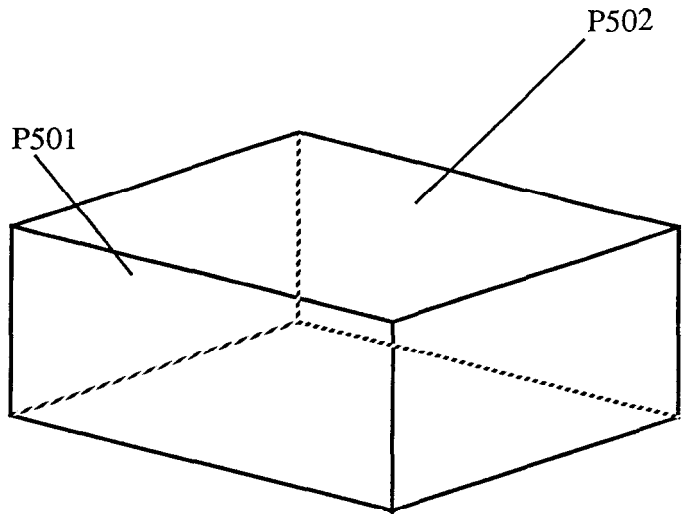


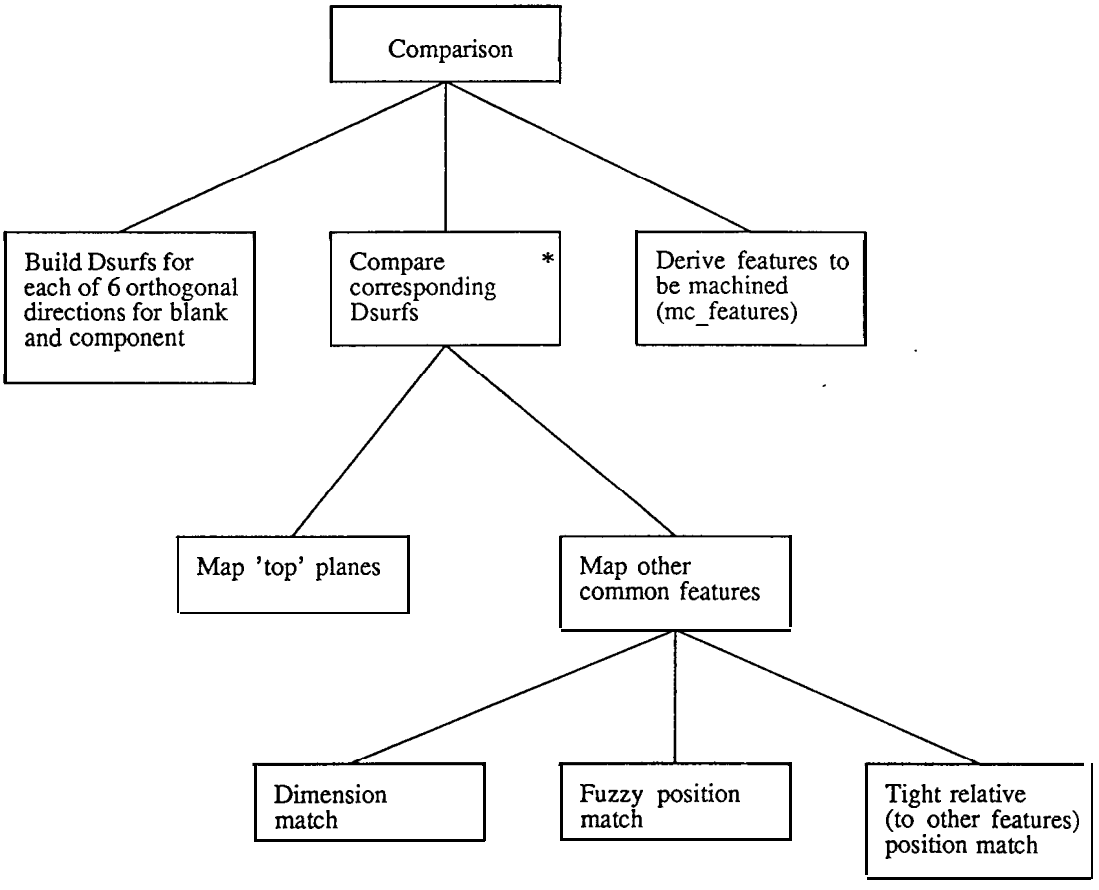
Figure 3.6: Simple blank

3.5 Plan Space Generation

The overall architecture of the system is shown in Figure 3.8. The planning process is controlled by the planning engine which makes use of separate inference engines for handling frames and production rules, the means by which the manufacturing knowledge is represented. The work of the planning module is to make use of this knowledge to discover candidates for, constraints on and interrelationships between the operations to be used to manufacture the component features. This task involves reasoning about the part and the capabilities of the machines. Hence the knowledge bases are used to reason over the integrated networks used to describe the blank, component and machine shop.

A high-level description of the basic plan space generation algorithm, particularly how it builds and manipulates planning networks like the one shown in Figure 3.2, has already been give in Section 3.2. Further details will now be given. The planning networks are represented explicitly using the following prolog data structures:

```
sub-net (N ,Ft ,
   Ft finished-with <data . . .>
& <data . . .> processed-from Ift
&( Ift roughed-with <data1 . ..>
    or Ift roughed-with <data2 . ..>
```



NOTE: This is a JSP hierarchical decomposition algorithm diagram. Such diagrams are used fairly regularly in this thesis. Read left to right for sequence of routines; * denotes an iterative process; o denotes a conditional branch.

Figure 3.7: Comparison algorithm

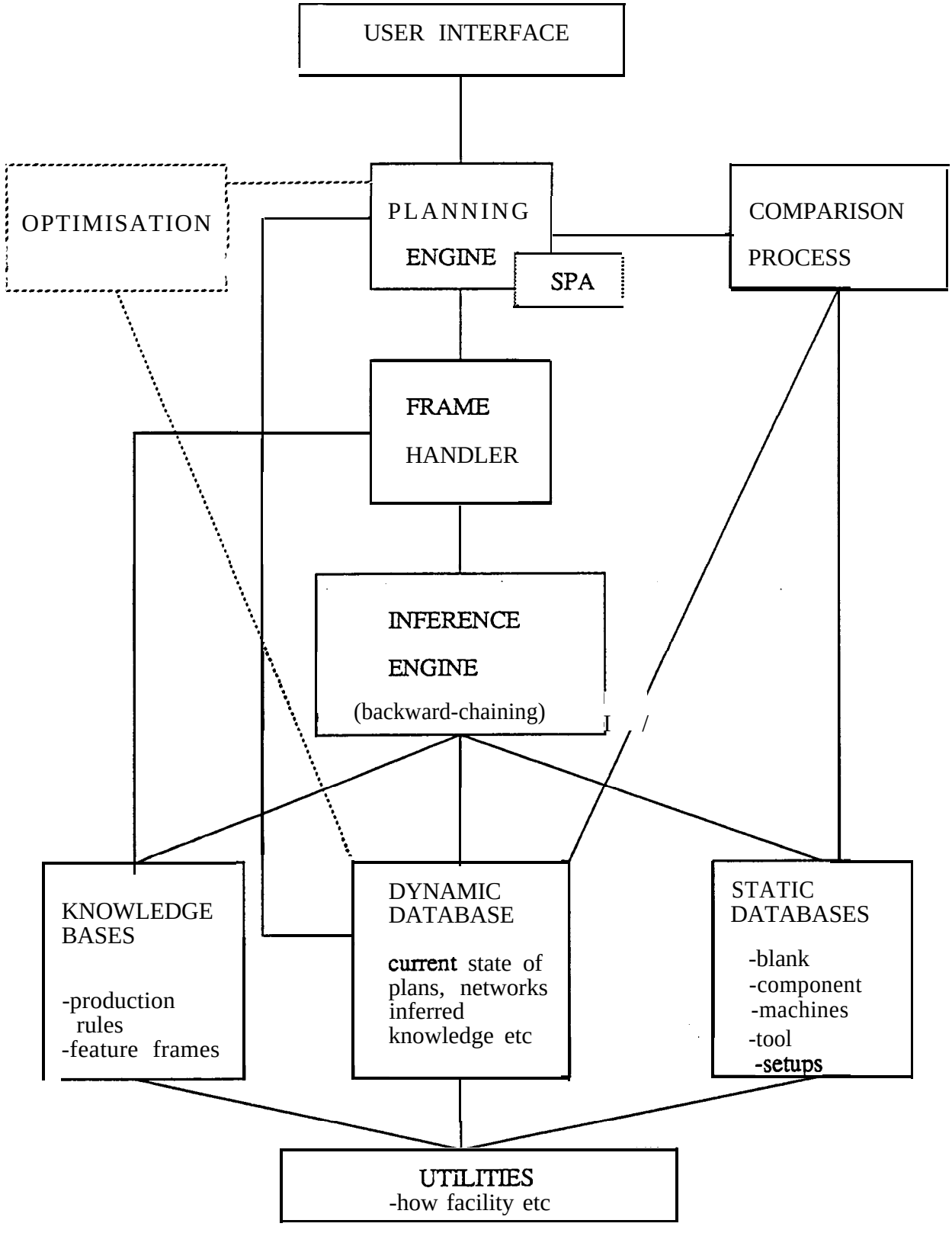


Figure 3.8: System architecture

```
    or . . .
  )
  ).

net(N,Ft,Sn1 & Sn2 & Sn3 &...).
```

The 'sub_net' structure simply lists the binary relationships making up that part of the network emanating from a single finishing node. The 'net' structure pulls together all the subnets corresponding to the same feature. The & is a. defined infix operator. The <data . . .> symbols stands for the information inferred by the planner and put, into the process nodes. Ft stands for the final feature and Ift for an intermediate feature. N is instantiated to a number which uniquely labels the structure. The reasons for this organisation and the details of the contents of <data . . .> nodes will be made clear in the next section where the planner's reasoning is described.

The central planning algorithm is outlined in Figure 3.11. The planning cycle starts with the detection of an unplanned deep-feature which has been labelled at the comparison stage as needing machining. A deep-feature is simply one with no sub-features. This allows backward planning from the deepest parts of the component out to the surface. Referring back to Figure 3.8, the planning engine takes care of the high level planning, manipulating the planning networks. The planning engine looks for strong interactions each time a new feature is presented for planning. Such interactions would include a feature with a very tight parallel tolerance with respect to some other feature. If a strong interactions is found, using an appropriate rule base, the feature may well have to be deferred for later planning. For instance, two or more features with tight parallel tolerances to each other must be finished on the same setup. Hence they must be planned as a group and the networks built accordingly. The algorithm employed for this particular type of interaction is shown in Figure 3.9. It makes use of many parts of the system involved in the core plan space generation algorithms.

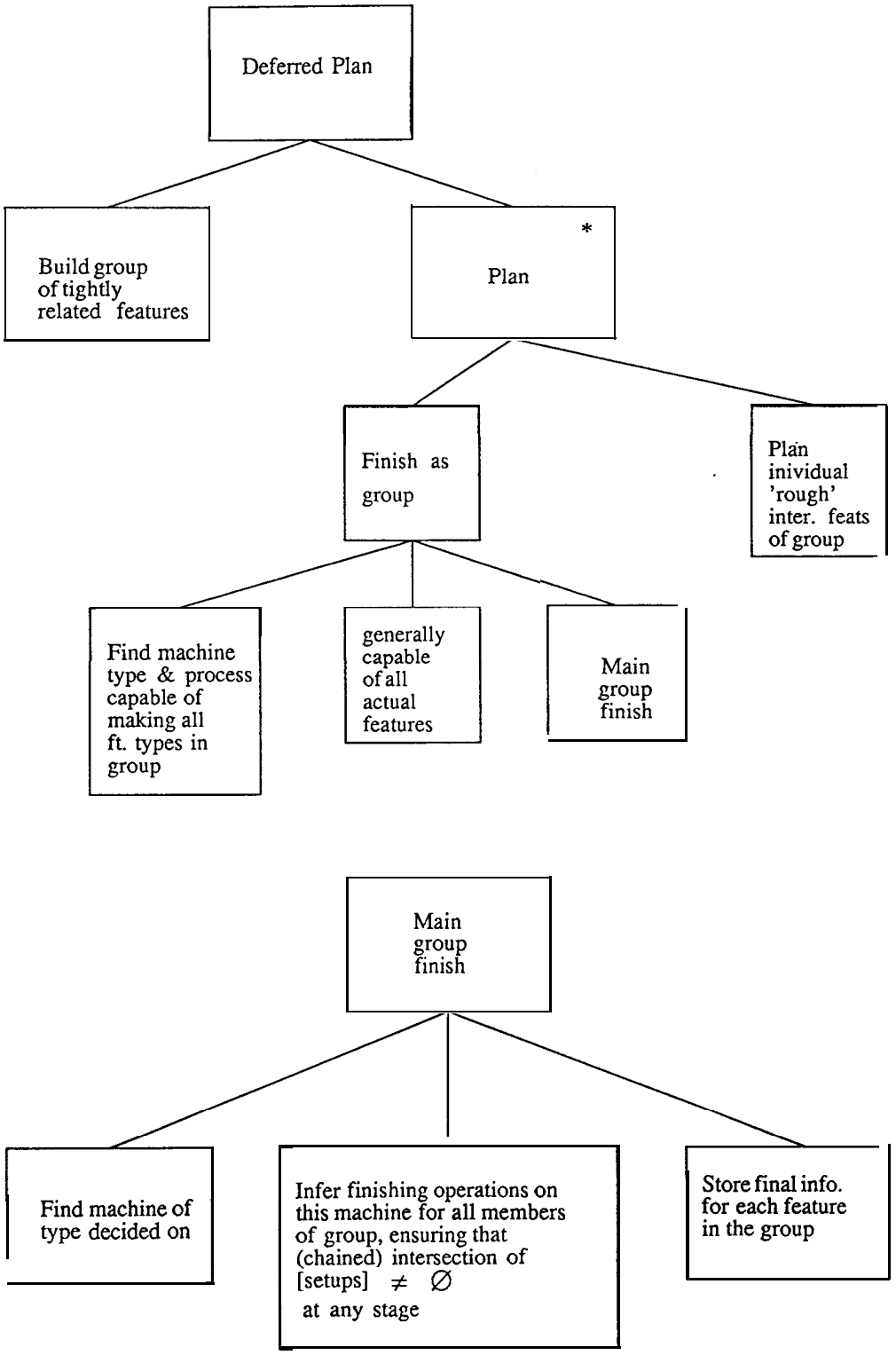


Figure 3.9: Deferred planning

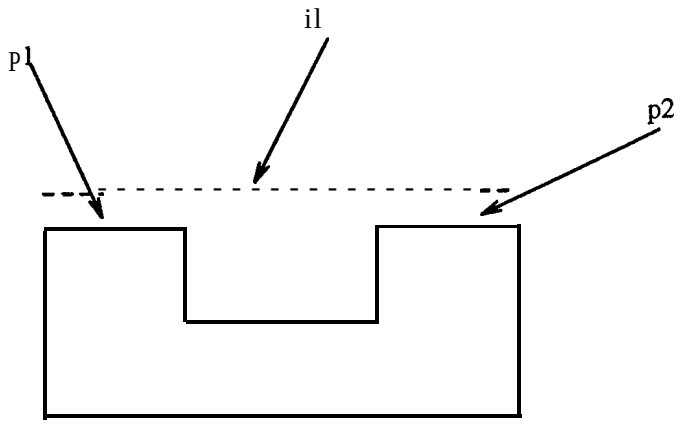


Figure 3.10: Tagged planes.

Another case handled by special purpose algorithms is that of separate surface planes emanating from the same intermediate feature. An simple example is shown in Figure 3.10. Here, in the final component, pl and p2 are separate planes (quite possibly with different properties such as surface finish) broken by the through slot. However, they lie in the same plane and would have come from the same intermediate feature (il in the diagram). Such an intermediate feature is referred to as virtual, since it does not have such a direct relation to the component feature as in the standard case. The algorithm used to handle this situation is given below.

1. Collect surface planes tagged as coming from some (virtual) intermediate feature.
2. Merge planes to create the intermediate feature.
3. Find the feature in the tagged group with the tightest surface finishing tolerance.
4. Backwards plan each individual tagged feature (using the standard algorithms), but force all to use the roughing operations (on their common intermediate feature) used by the feature found in 3.

In this case, as in the deferred planning case, the networks and subnets built up must be linked appropriately to reflect the dependencies inherent in these special situations.

If there are no strong interactions of this type the planning engine passes control down to the frame engine. The frame engine manipulates plan schemas which are the highest level static knowledge source used to fill up the <data . . , > nodes. A simple plan schema is shown below:

```
mframe(1,thru_slot,F,F,
do (rough F),
(acons(rough top of F),
   act(F roughed-on _ using _ tool _ using _))
).
```

This tells us that in order to find a roughing operation for some through slot, F, we must look for a rule with an action part matching the 'act' slot. It also tells us that an anteriority, or ordering, constraint must be laid down: whatever is at the top of F must be roughed before F is roughed. Some of the schemas used are more complicated involving conditional slot contents and more slots. They allow the description of some important constraints, such as orderings, to be stored in the knowledge base rather than being embedded in the reasoning software. There are other, more complex, frames which contain information on how to deduce geometric properties of a feature or its surroundings, for instance how to deduce what is on 'top of' a through slot. In all, more than 50 frames were used in the implemented system. Once the frame engine has decided on a specific set of information to deduce, it hands over control to the backward chaining inference engine that handles all the production rule bases. A simple rule, which covers the basic syntax, is shown below:

```
rule(216,mcsel,thru_slot,
if
  current_active_fin(Ft,Mcf)
& Mcf min_surf_req M_s
& Mtype makes thru_slot using Process
& Mcr isa Mtype
& not current_active_rgh(Ift ,Mcr)
& Mcr surftol Mtol using Process
& Mtol less M_s
then
  Ift roughed-on Mcr using Process).
```

This rule itself would have been called up in an attempt to prove one of the conditions of a higher level rule related to finding a. roughing process. In order to keep the search space as small as possible, the rule base is doubly partitioned. The first partition relates to the rule context or function, the second to the generic feature type involved. The rules use some important built, in functions. An example is 'currentactivefin' shown above. This function is a call from the inference engine up to the planning engine in order to find out which sub-net is currently active and hence which finishing node we are currently building a route from. The interaction between the finishing and roughing processes is clearly seen in the rule. The rules are able to give information on what [machine/process/tool/setups] are suitable for various operations, how much operations will cost and other manufacturing knowledge related to constraints. Note that many of the rule conditions are simple binary relations from the part and machine shop representations. Thus the building blocks of the highly integrated representations shown in Figure 3.5 can be used in a very natural way, as rule conditions, to provide a simple and powerful method of reasoning over the networks. The rule bases cover information on selcting machines and tools, finding setups and costing machining operations. In all, more than 200 rules were used in the implemented system.

The various engines also have to deal with more complicated planning involving

more direct interaction. For instance, the planning of a group of features, with very tight relative tolerances, must be deferred until the whole group has been found. As already mentioned, situations like this are spotted by the planning engine and then dealt with by using special case algorithms which make use of much of the machinery of the main planning algorithm.

As the networks are built up the part representation is updated to reflect the changes in geometry. An intermediate feature is formed from a finished-featrure (feature on component) by creating a new feature of the same type from a standard template and then deriving the intermediates feature's properties from those of the finished-feature. Some of the information will be the same for both features (e.g. centre coordinates for a hole) while much will be slightly different (e.g. the radius of an intermediate hole will be slightly less than for the corresponding finished-feature hole; the surface finish of the intermediate feature will usually be greater (higher tolerance) than for the finished-feature). The exact transformation of the geometric, and other, properties will depend on the [machine/tool/process] combination used to finish the feature. After the roughing operations have been found the intermediate feature is 'filled in' in the model.

To summarise, the planning algorithm attempts to break the manufacture of the component down into a number of nearly independent stages. It does this by looking out for possible interactions and then using special purpose algorithms to deal with them. This is based on Sacerdoti's idea of critics [97]. The entire space of possible plans can then be generated by finding all the possible operations to carry out each step along with the ordering constraints which must exist, between the steps. Essentially a step refers to a finishing operation on a single feature or a super-feature (a group of features treated as one due to some network of constraints binding them together) or a roughing operation on an intermediate feature. So each step of the plan has a unique feature, super-feature or intermediate feature associated with it. The operations found to manufacture these are described in terms of [machine/process/tool/setup/cost] combinations. The setup refers to the orientation of the workpiece and the cost refers to the machining cost associated with that

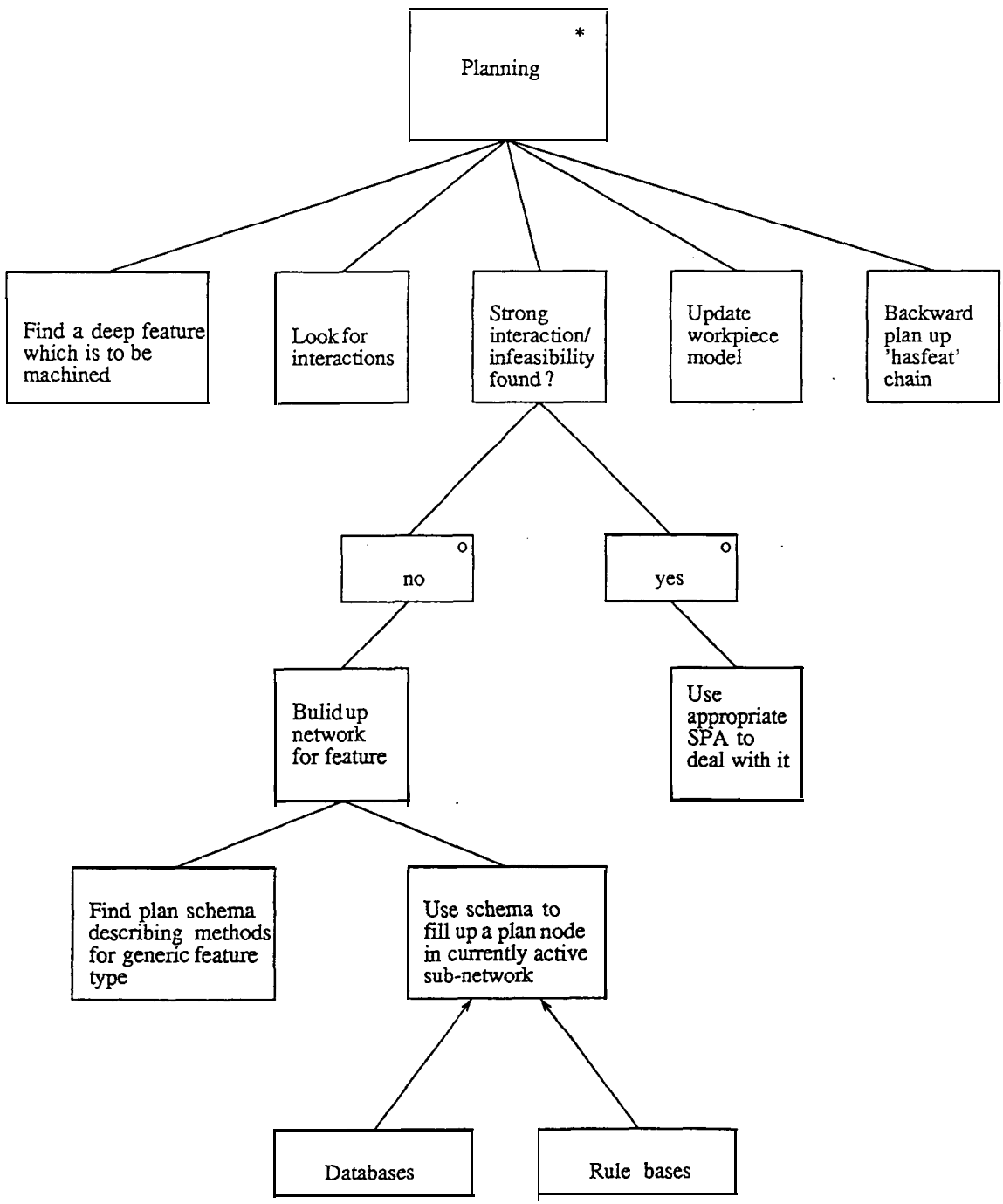


Figure 3.11: Central plan space generation algorithm

operation. Along with this information the planner generates a separate network representing the partial orderings it has deduced hold between the steps of the plan. Such a network is shown in Figure 3.3.

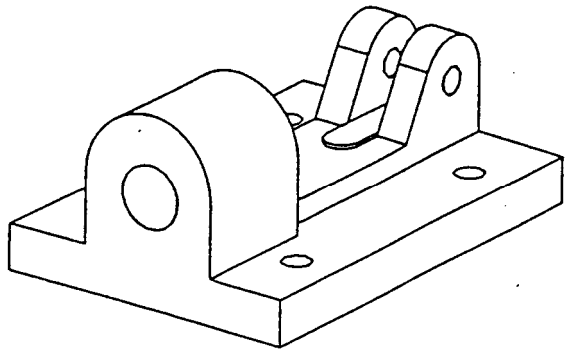
A sample of components modelled using the feature-based representation language and successfully handled by the plan space generator are shown in Figure 3.12. Some are specially designed pathological examples with nested features and many inter-feature constraints, these were used to stretch the algorithms and test the feasibility of the approach. Others were actual components [e.g. the toggle clamp shown in the Figure) manufactured by various companies. Graham Pedley was responsible for modelling most of the real components [90]. A variety of other components not, shown were also used. The level of complexity of components shown in the Figure is representative.

The output of the plan space generator, which acts as the interface to the GA optimisation module, is briefly described in Appendix A.

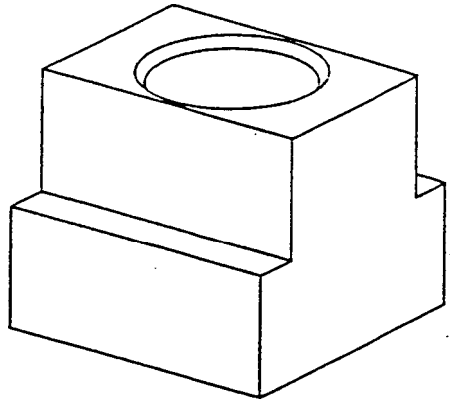
3.6 Later Object-oriented implementation

The part representations used in the earlier representation system are quite verbose and hand coding is very tedious. This means that automatic generation is a necessity for a commercial implementation. Feature information could either be extracted from a solid model of the component or, more promisingly, be generated directly by a feature based design system. Work in that direction may encourage a tighter integration of a design and planning system.

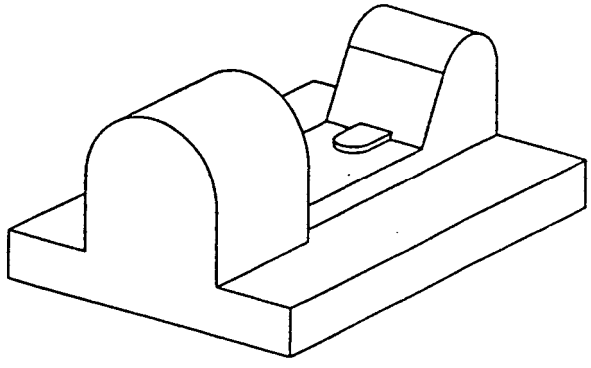
This approach was taken by a follow-up SERC project to the one in which the bulk of the work reported here was done. A later C++ implemetation of the plan space generator is linked directly to a feature based design system [30]. This is a class based object oriented version. The component model is passed down by the design-system along with information flagging potential problems with tool access and so on. These pieces of information are derived from the solids model maintained by the design



WDS Component 4931.U. : Toggle Clamp Base
Finished Component



WDS Component 664-210 : T-Nut



WDS Component 4931M : Toggle Clamp Base
Initial Casting

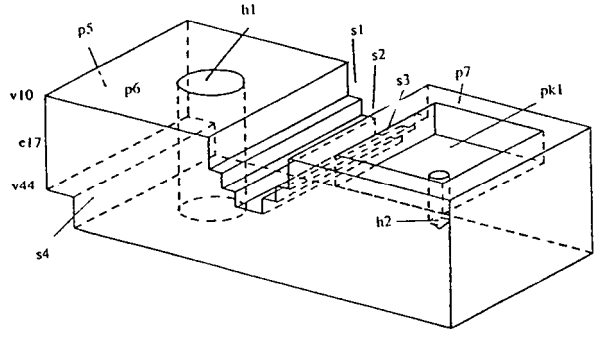


Figure 3.12: Example components used.

system. The whole solids model can be passed down to the plan space generator potentially allowing advanced geometric reasoning. Features, machines, processes, and tools, are now represented in explicit class hierarchies. The core mechanism for process, machine, tool and setup selection is as follows. Feature objects have a list of possible manufacturing processes associated with them; process objects are interrogated by the feature objects to see they if they are suitable. Process object have a list of machine types usually capable of performing them; process objects interrogate the machine class to find actual machines that can be used. In a similar way the tools and possible setups for a given [feature / process / machine] combination are found.

A sketch of the class hierachies used in this version of the system is shown in Figure 3.13. Each class defines objects (features, machines, tools etc.) in terms of a set, of members. Some of these are simple numerical fields holding the value of a tolerance or a dimension or some such. Some are lists of, for instance, possible manufacturing processes that can be used for a given feature type. Some are complex functions that interrogate other class objects and do a job such as computing a manufacturing cost, or testing the viability of some manufacturing process for a particular feature on the component. In this way, the manufacturing data and knowledge is represented in a more implicit and less user-friendly way than in the data-bases and rule-bases of the earlier implementation. The advantages are flexibility – particular knowledge representation formalisms do not have to be adhered to, algorithms encapsulating manufacturing knowledge are implemented in raw C++, and speed – the C++ version is at least two orders of magnitudes faster than the PROLOG implementation. The disadvantage is that it is harder for a potential user of the system to model their own manufacturing facility and methods.

The core algorithm of the later implementation is shown below. It can be seen that this is not quite the same as the earlier algorithm, but it is nearly functionally identical. Again intermediate features are created from features (or other intermediate features) according to the last manufacturing operation to be performed on the parent feature. This allows the generation of finished-feature manufacturing routes of

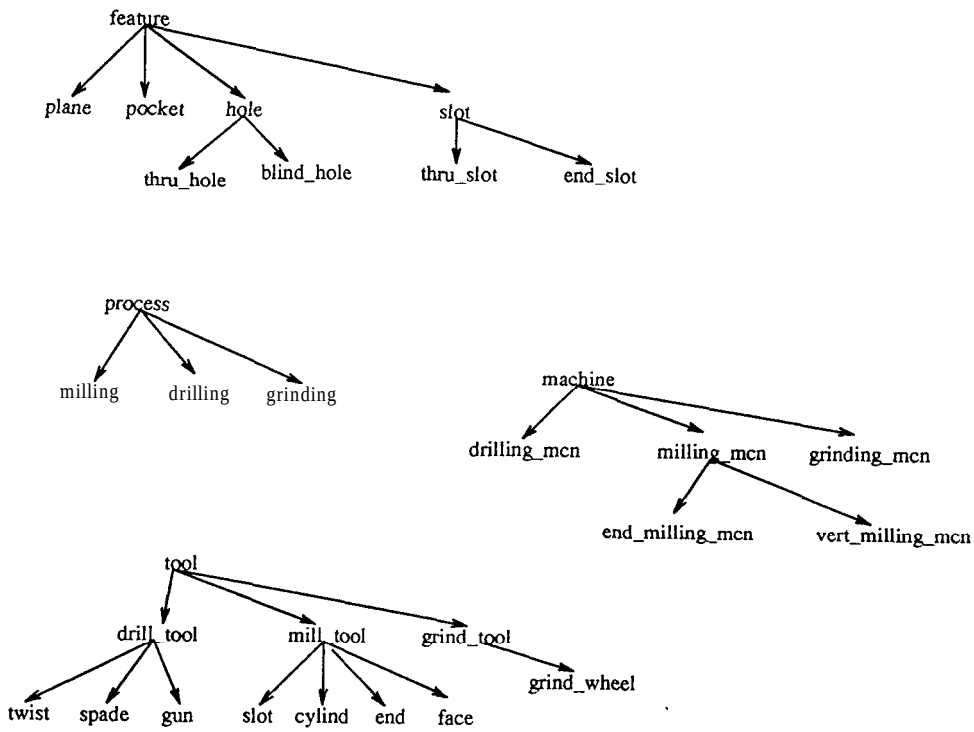


Figure 3.13: Class hierarchies.

any length, containing any number of finishing and roughing operations. The only constraint in choosing the next operation, is that it must make some progress over the last operation. Since we are backwards planning, this means it must leave an intermediate feature which is closer to a totally 'filled-in' (or NULL) feature than the last one. The 'hasfeat' relation of the earlier implementation has here being replaced with explicit 'parent' and 'child' relationships. **Plan_Space()**

1. Read in component and blank models, a list of features to be manufactured **(FeaturesToBeManufactured)** is passed in as part of this; read in machine shop model.

2. i = 0, pass-flag = 1.

3. Ft = FeaturesToBeManufactured[i], If at end of list Go to 16, Else If Ft has not been processed and is not deferred and, If **pass_flag=2,** Go to 4, Else if Ft is a deep feature Go to 4. Else Go to 14.

4. If Ft interacts strongly with other features (according to any of the special purpose tests), append it to deferredlist, Go to **14.** Else Go to 5.

5. Unless Ft is an intermediate feature, create a **dataNode** object for storing manufacturing information. j= 0.

6. Find the jth possible Process in Ft's list of potential manufacturing processes. If at end of list, Go to 14, Else k = 0.

7. Find the kth possible machine from the list attached to the Process found in 6. If at end of list Go to 13, Else If this machine is generally capable of the job Go to 8, Else Go to 12.

8. If a tool can be found for the current Feature/Machine/Process combination, Go to 9, Else Go to 12.

9. If the resulting Feature/Machine/Process/Tool combination can be costed Go to 10, Else Go to 12.

10. Create a SubNode to the current dataNode (or current SubNode if Ft is an intermediate feature) and store in it the resulting Feature/Machine/Process/Tool/Cost information.

11. Generate an intermediate feature (Ift) from current feature (Ft), Unless Ift is NULL (chain of operations complete), build next link in chain by (Recursively) setting Ft=Ift and Go to 4.

12. (Unwind recursion one level), k = k + 1, Go to 7.

13. j = j + 1, Go to 6.

14. If Ft has any unplanned 'parents', set Ft to next one and Go to 4, else Go to 15.

15. i = i + 1, Go to 3.

16. pass_flag=2.

17. Plan deferredlist.

18. Exit.

To give an idea of the complexity of the classes, part of the ManuFeature class (describing features) is shown. Many of the members are complex functions.

```
class ManufFeature : public Thing{
public:
ManufFeature(int, char*, int );
ManufFeature();
~ManufFeature();
virtual void display(){} //display to screen
void setMadefl(int );
void setIntmfl(int );
void set_FtName(char* );
void set_FtType(char* );
char *give_FtType();
void set_position(double, double, double );
void set_orientation(double, double, doubia );
void setFtId(int );
void set_isDeep(int );
```

```
void set_isDeferred(int );
void set_parents(char** );
void set_children(char** );
int give_isMade();
int give_isIntermediate();
int give_isDeferred();
char *give_ftName();
int FtId();
int give_isDeep();
char *give_child(int );
char *give_parent(int );
void set_chpt(int ,ManufFeature* );
void set_pntpt(int ,ManufFeature* );
ManufFeature *give_pntpt(int );
char *give_process(int  );
virtual int Sinteracts(){return 0;};
int *give_setups();

virtual int mcntol_fit(Machine* ){return 0;};
virtual int proctol_fits(Machine* ,Process* ){return 0;};
virtual int tool_tol_fits(Tool* ){return 0;};
virtual float cost(Machine* ,Process* ,Tool* ,int ){return 0;}
virtual int setups(Tool* ){return 0;};
virtual float give_surfFin(){return 0;};
virtual float give_minsurfFin(){return 0;};

protected:
int isMade;    //already planned flag
int isDeep;    // is deep feature flag
int isIntermediate;   // is intermediate feature flag
char *ftName;      // name
char *ftType;      // type
double x;          //coordinates
double y;
double z;
double yaw;    // orientation
double pitch;
double roll;
int featureId;   // ID
char *manProcs[6];  // possible manf. processes
int isDeferred;     // deferred planning flag
char *DefType;     // why deferred
char *children[12];  // list of child features (has-feat)
ManufFeature *chpt[12];// pointers to these
char *parents[12];      // likewise parent features
ManufFeature *prnpt[12];
int *Sups;        // list of setup codes
       };
```

3.7 Research issues and Assumptions made

The whole methodology described is crucially dependent on being able to break the manufacturing process down into nearly independent stages so that an exhaustive set of possible plans can be implicitly generated in polynomial time. Questions of stability, clamping, fixturing and tool access have not been fully tackled yet. In particular their impact on the methodology needs to be thoroughly investigated, a more hierarchical approach may well be needed.

At this point it it useful to draw together the basic underlying assumptions made throughout. They are:

- The tool selection aspect of the problem is ignored. Tool change costs are effectively zero.
- Machine transfer costs are ignored. The physical layout of the job-shop is not. taken into account.
- It is assumed that manufacturing requires at most one roughing and one finishing operation per feature. In principle it would be straightforward to lift this simplifying assumption.
- Selection of fixtures, and the impact of clamping and fixturing issues on the feasibility of using a particular machine on any given feature, is ignored.
- Tool access checking has not yet been integrated into the system.
- Complex planning issues relating to geometrically interacting features, thin walls, and the like are not dealt with.

Some of these issues would be straightforward to incorporate (tool costs and machine transfers), while the rest are very challenging problems in automated process planning. However, complex planning decisions could be made manually. Such manual inputs to the system would impose constraints on the plan space to be searched by the GA, in terms of machine choices for particular features or combinations of

features, operation orderings and so on. Such a semi-automated approach should fit fairly comfortably into the general framework presented in this thesis.

3.8 Summary

This chapter has described a set of algorithms used to generate an exhaustive set of process plans given a feature-based description of the desired component and the blank, along with a model of the machine shop. Two implementations of this plan space generator were discussed, one in PROLOG and one in C++. The plan space generated consists of interconnected networks describing dependencies between the operations in a plan. An exhaustive set of alternative machine/setup combinations for each operation are listed. A separate network holds all the ordering constraints between the operations. These constraints are generated by the planner. This space is intended to be searched by an optimisation technique to find low cost plans according to some realistic criteria. This aspect of the research is discussed in the following chapters. Finally, the assumptions made in this work were drawn together and made explicit.